IVI-EG 03

21.Jan.2021

Table of contents

Discussion Plan (minor update)

- Plan

#	date	Discussion Topics
1	Dec. 8, 2020	Kickoff, LifecycleManagement,
2	Jan. 7, 2021	LifecycleManagement, HelathMonitoring, + *“HAL”, Yocto Recipe Commit*
3	Jan. 21, 2021	~~HelathMonitoring, PowerManagement,~~ **Commit Review, HAL(CAN)**
4	Feb. 4, 2021	PowerManagement, +(Quick introduction to TestFW from Jan-Simon)
5	Feb. 18, 2021	TBD
	…	
	TBD (within trial)	Agl TestFW adoption Error Management / Logger service DEMO/Presentation for AMM

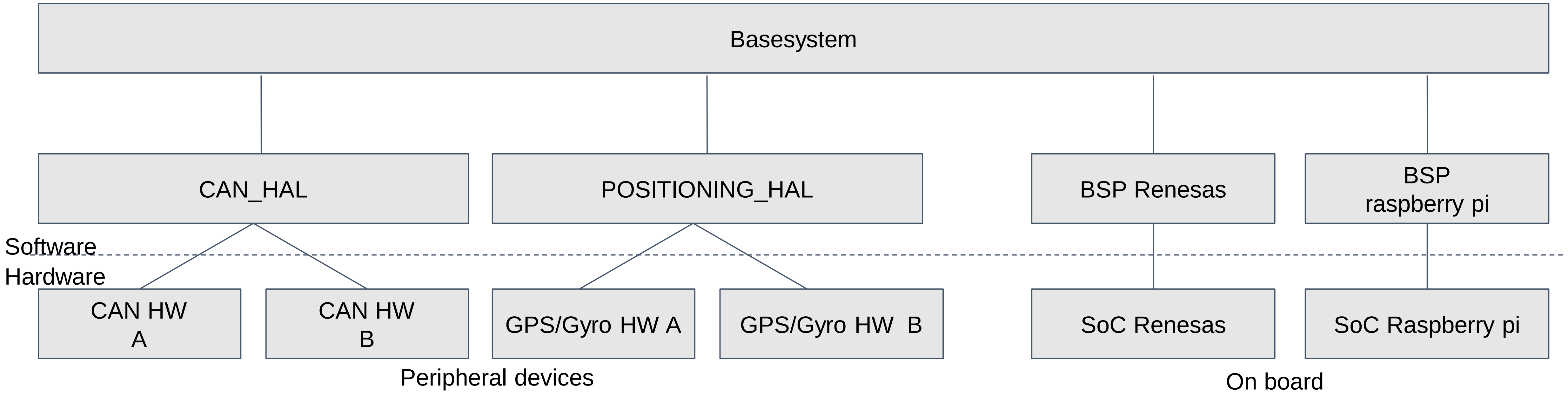
- No other update.

About Commit Review

- 25646-25652 review has been done without explaining HAL API.
- 25653-25654 review hasn’t been done yet.
    - https://gerrit.automotivelinux.org/gerrit/c/AGL/meta-agl-devel/+/25653
    - https://gerrit.automotivelinux.org/gerrit/c/AGL/meta-agl-devel/+/25654

What is HAL?

1. What is basesystem “HAL”?
   a. Abstracted programming interfaces when basesystem access to hardware resources
   b. Sample implementations and stubs are included in currently disclosed basesystem.git
2. What is the difference between BSP and HAL?
   a. BSP is for SoC.
   b. HAL is mainly for Peripheral devices.

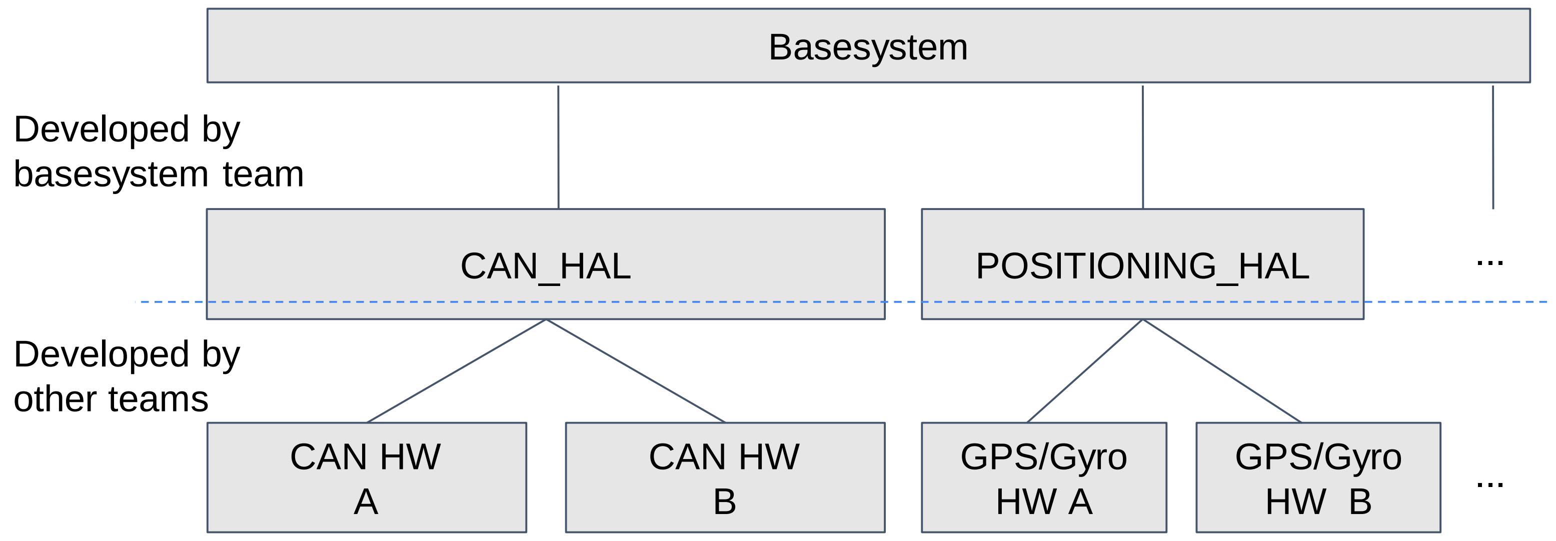


HAL:
boot, **can**, clock, deck, input, nv, **positioning**, power, security, soc_temperature, usb, vehicle, video_in

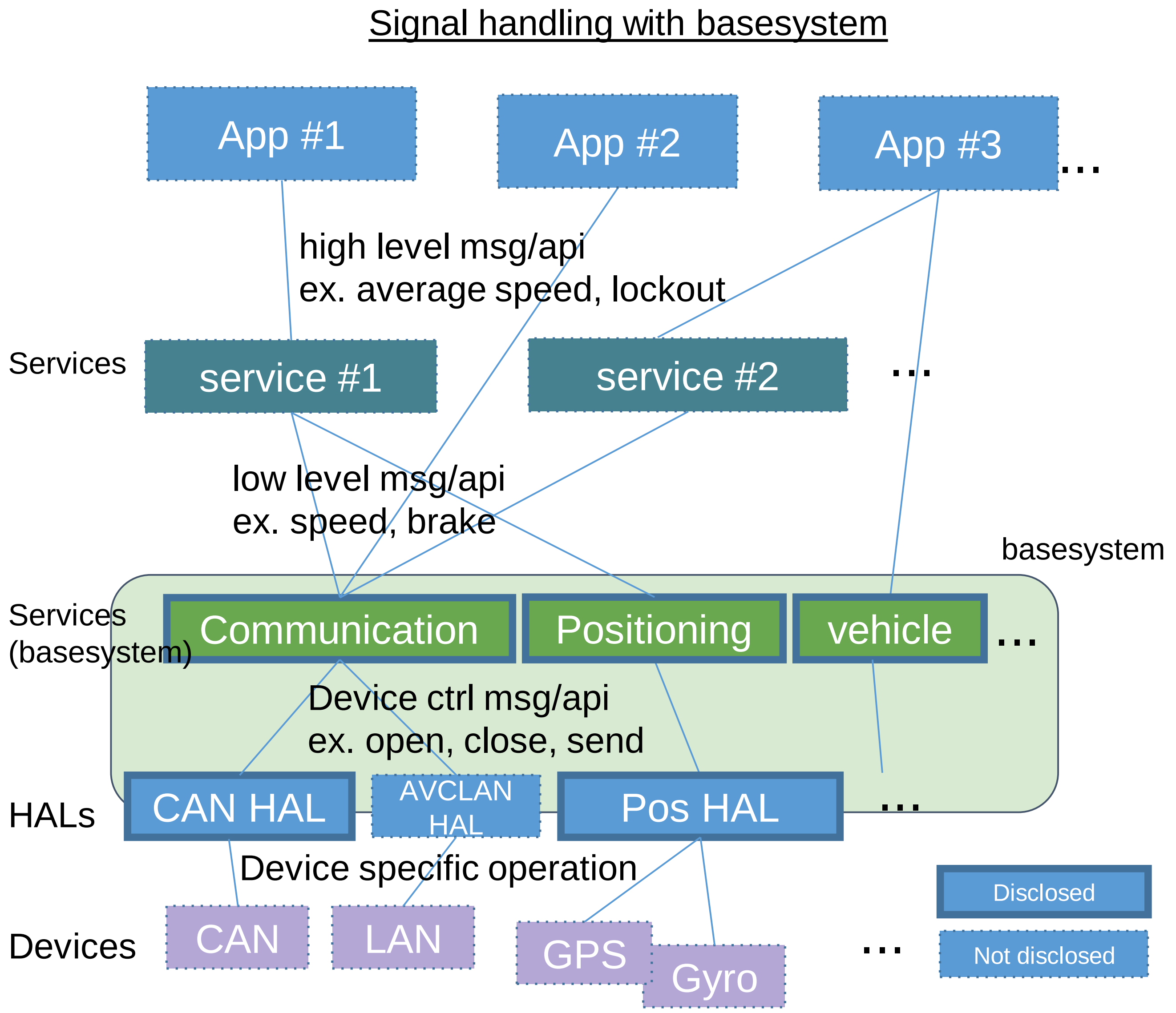
Why is HAL needed for the product?

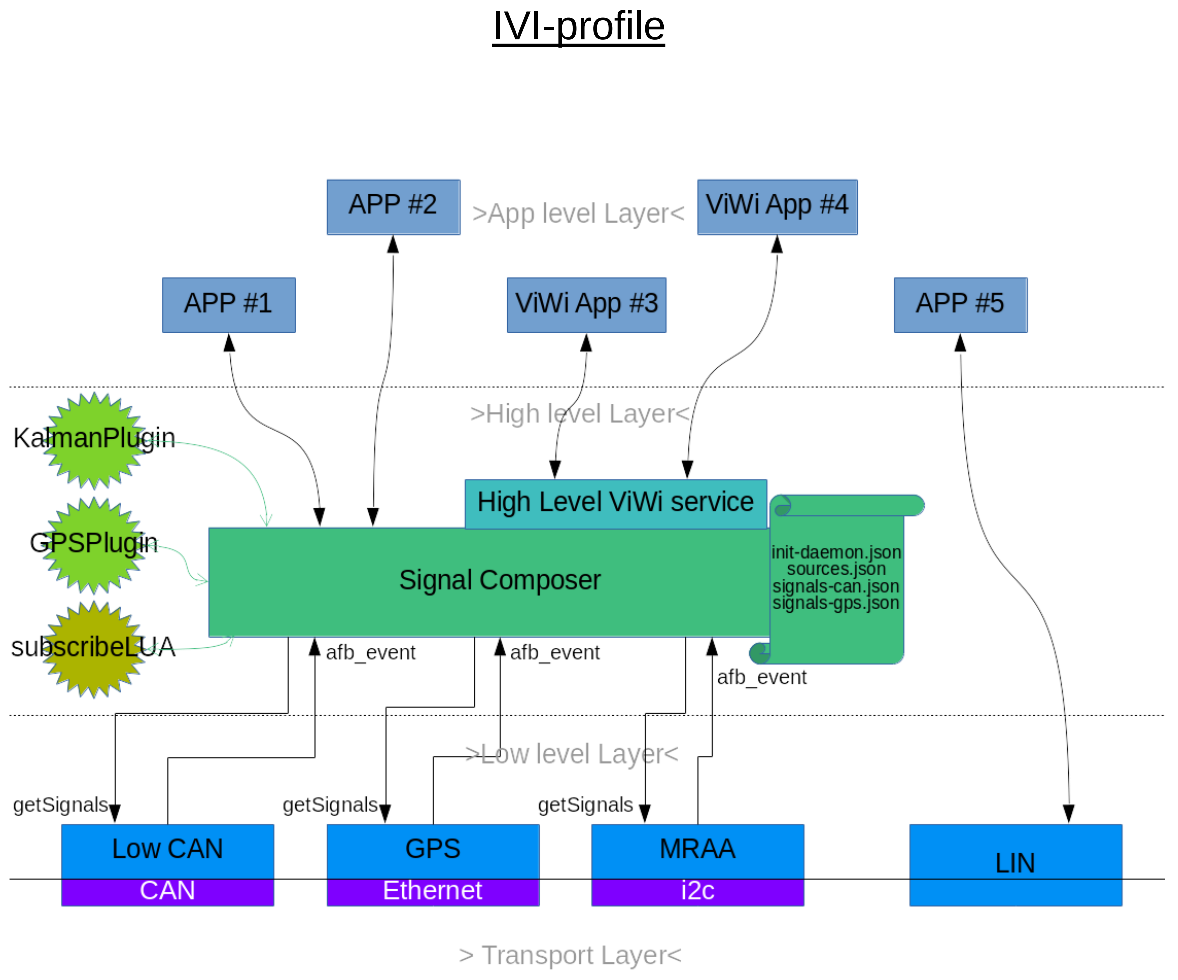
1. Why is HAL needed?
   a. By implementing HAL, various devices can be supported without changing basesystem and upper services.
   b. Application can be developed and tested before Hardware is prepared
   c. Multiple teams (suppliers) can develop different layers independently.

1. How did you define the boundary surfaces and I/Fs?
   a. Common functionality vs HW specific
   b. Historical and organizational reason. Not architecturally optimized.



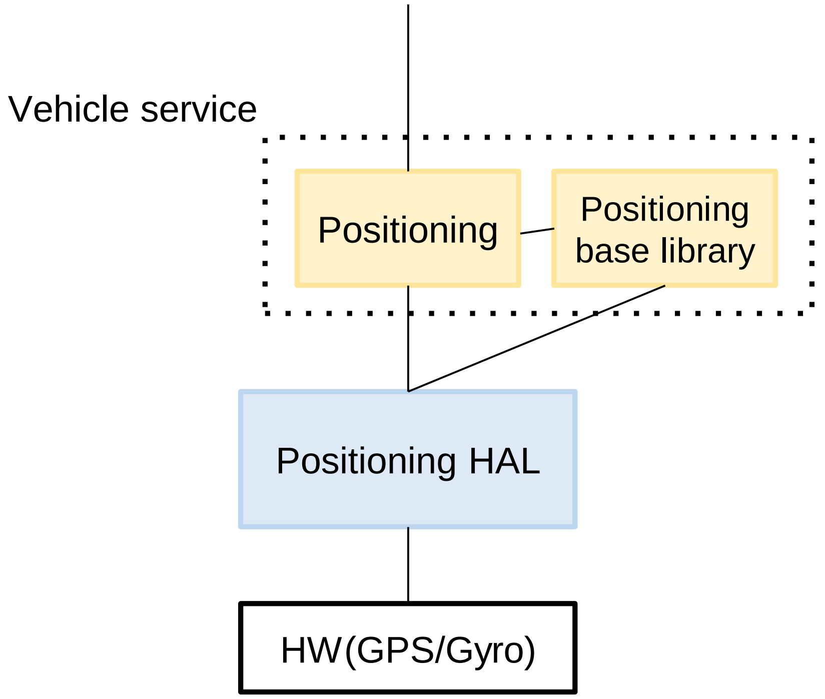
Signal handling overview





HAL Examples 1: Positioning HAL

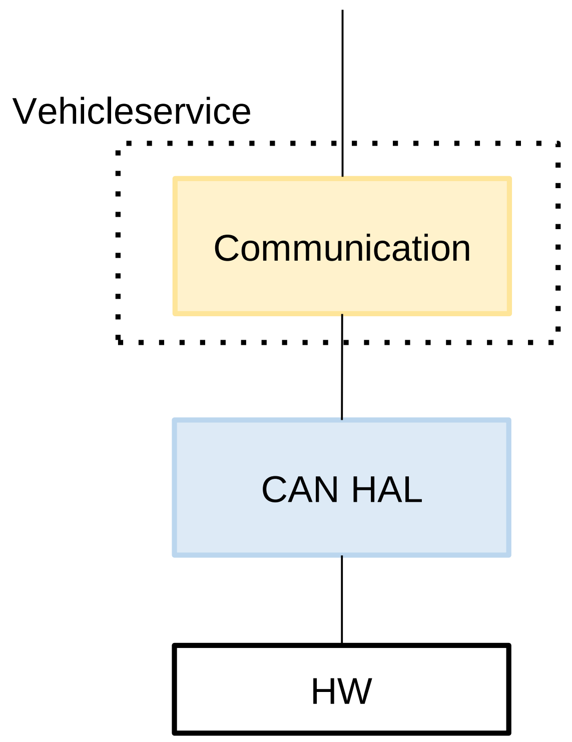
- Positioning HAL is the HAL which supports following features
  - Receive data from GPS chips and analyze, and provide GPS data to UI.
  - Receive Gyro sensor data from each devices and provide them to UI.
- The functions detail
  - Receive GPS data from GPS chips and provide the data
  - Provide position information (longitude, latitude, altitude, GPS time, direction etc.)
  - Set GPS time
  - Request GPS chips reset
  - GPS receive error
  - Notify GPS week counter
  - Provide raw GPS time
  - Provide Information about Gyro Sensor



Positioning HAL Software block diagram

HAL Examples 2: CAN HAL

- CAN_HAL is the HAL which supports implementation of CAN protocol stack.
- By receiving a request from “Communication” which is one of Basesystem units, it sends the CAN data to the upper layer.
- These are CAN_HALfunctions.
  - Initialize and finalize each communication path
  - Send CAN frame, get the status and send it back to the sender(communication)
  - Receive CAN frame from the CAN device
  - Get CAN micon version



CAN HAL Software block diagram

Why not vcan / socketCAN?

1. Some product specific requirements need the support from Micon.
2. It’s not easy to meet these requirement with socketCAN (for now).

	basesystem (+ an **example** of Product)	IVI-Profile
COM (normal traffic)	[Communication Service(CAN)] ● Send / Receive (subscribe) ● Echo back ● Com Watch (timeout call back) ● CAN service availability	[can-low-level, can-high-level?]
“HAL I/F”	[can hal] ● Open / Close / Send / Recv ● Micon ver get	(can-low-level)
Network Management	[Micon + HAL implementation] ● ex. AUTOSAR NM (sleep /wkup)	?
Diagnostic	[Micon + HAL implementation + Diag service] ● ex. UDS, OBD	[can-low-level?]
CAN Driver	[misc (basesystem doesn’t care)]	[socketCAN]

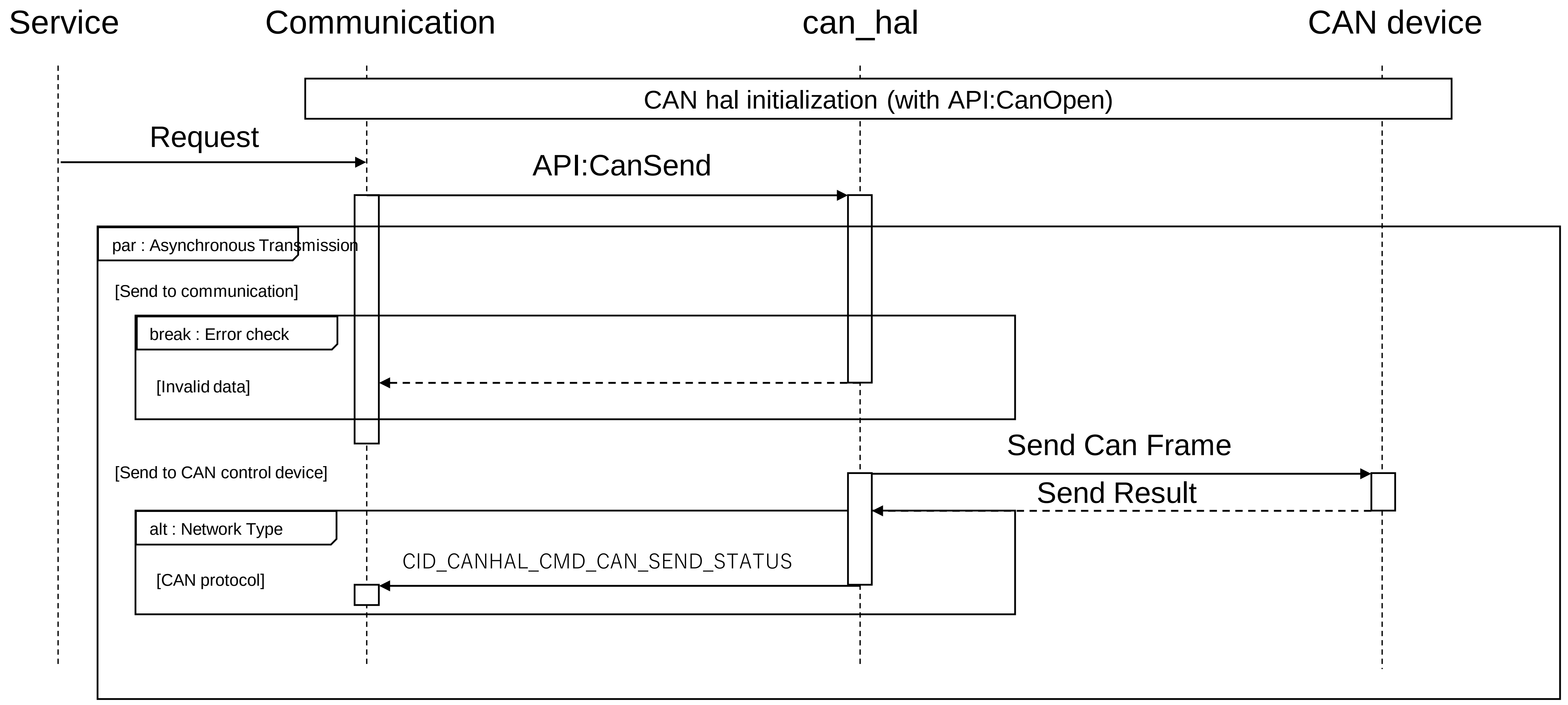
CAN HAL API / Command

API name	Description
CanOpen	API to initialize each communication path(CAN or other protocol), which needs CAN type(protocol) and application handle as argument.
CanClose	API to finalize each communication path(CAN or other protocol), which needs CAN type(protocol) and application handle as argument.
CanSend	API to send CAN frame which needs pointer to message data, CAN type and application handle as argument.
CanGetVersion	API to get Can microcomputer version information which needs version buffer and application handle as argument.

Command	Description
CID_CANHAL_CMD_CAN_READY	Notify availability of Global CAN.
CID_CANHAL_CMD_CAN_SEND_STATUS	Notify send result of CAN
CID_CANHAL_CMD_CAN_RECV	Notify receive of CAN

Detail : staging/basesystem.git; hal/can_hal/hal_api/can_hal.h

Typical use case : Send CAN data



Plan (Idea) for Production Readiness

- We have disclosed HAL APIs, and HAL implementation is sample only.
- No future plan to contribute the HW specific implementation.
  - It’s related to HW support of Production Readiness.
- We think it’s unlikely to merge HALs into IVI-Profile as is.
- Still, we think current HALs can be the reference of actual products.